5주차 결과보고서

전공: 수학과,컴퓨터공학과 학년: 3학년 학번: 20181256 이름: 김도현

1.1

실습 1은 4주차 실습 코드가 들어있는 폴더를 로컬 저장소로 만들고 이후 원격저장소에 push 하는 것이다.

git init 을 하여 로컬 저장소를 생성했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

다음은 실제 원격저장소에 업로드할 파일들이 들어있는 로컬 저장소 폴더이다.( 사진에서 볼 수 있듯이 init을 통해 .git 숨김 파일이 생겼다. (실습 제출물1)

텍스트이(가) 표시된 사진

자동 생성된 설명

원격 저장소 repository가 comsil\_5\_github 여서 git remote add origin 을 통해 로컬, 원격 저장소를 서로 연결했다. 그리고 git remote –v로 정상적으로 연결했는지 확인했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

현재 파일이 아직 반영이 안되어 있음을 git status 로 확인하고 gid add\* 로 모든 파일들을 추가했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

git commit -m ‘실습 1’로 commit 하여 변경 내용을 새롭게 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이후 git push origin master를 통해 master branch의 commit 내용을 원격저장소(comsil\_5\_github)에 올린다. (실습 제출물2)

텍스트이(가) 표시된 사진

자동 생성된 설명

1.2

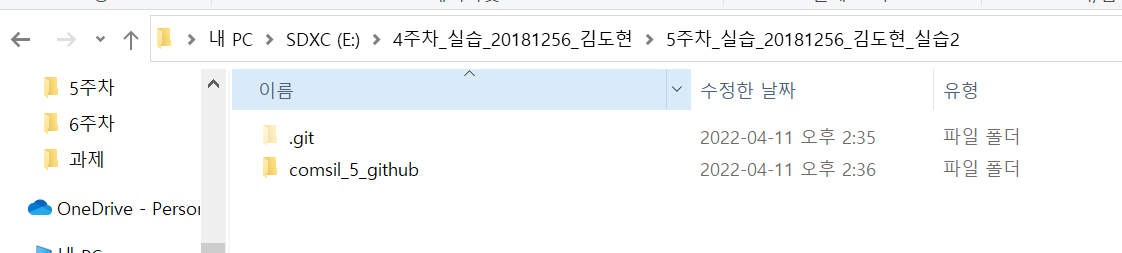
실습 2에서는 실습 1의 원격 저장소를 새로운 로컬 저장소에 복사하고 새로운 파일을 추가 후 다시 원격저장소에 push 한다.

새로운 폴더를 생성하고 git clone을 통해 원격저장소의 내용을 로컬저장소로 가져온다.

텍스트이(가) 표시된 사진

자동 생성된 설명

로컬 저장소에 정상적으로 가져왔음을 알 수 있다. (실습 제출물3)



ex1.txt ex2.txt ex3.txt 새로운 파일 추가하였다.

텍스트, 테이블이(가) 표시된 사진

자동 생성된 설명

새롭게 생긴 파일들을 git add \*로 추가하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

실습 1과 마찬가지로 실습 2에서 역시 git commit -m ‘실습 2’로 새 버전으로 commit 시킨다.

이후 git push origin master를 통해 새로운 파일들(ex1.txt, ex2.txt, ex3.txt) 이 있는 commit 내용이 원격저장소에 업데이트 된다. (실습 제출물 4)

텍스트이(가) 표시된 사진

자동 생성된 설명

생성된 원격 저장소 주소는 이렇다. 들어가면 정상적으로 실습1,실습2를 했음을 알 수 있다.

<https://github.com/ddohKim/comsil_5_github>

2.

깃 허브는 어떤 파일이 변경되고 삭제 및 추가되는지 확인하고 여러 이용자들이 효과적으로 작업을 하기 위한 깃(분산 버전 관리 시스템)을 더 쉽고 누구나 이용할 수 있도록 GUI 서비스로 간주 할 수 있다. 즉 깃의 장점을 대부분 수용한다.

우선 협업하여 어떤 프로젝트를 진행 할 때 대부분 그 프로젝트는 엄청 클 것이다.(프로젝트가 작으면 직접 사람이 수정할 수 있겠지만 그것 역시 귀찮은 일이다.) 즉 여러 사람이 여러 코드를 공유하여 동시에 짜거나 또 그 누구는 그 코드를 확인하고 수정해야 한다. 이는 소스 코드 관리를 하지 않으면 어떤 코드가 최신이고 누가 어떤 코드를 짜고 있는지 확인하기 힘들다. 또한 코드의 이동 및 복사도 불편하다. 효율적이지 못하다. 이 경우 이전코드를 보려면 작성한 모든 코드들을 가지고 있어야 하기 때문에 메모리 측면에서도 비효율적이다.

깃허브에서는 어떤 장소이든 온라인을 통해 쉽게 코드를 push하고 또 받아와 프로젝트를 진행 할 수 있다. 누군가 코드를 수정하여 깃허브에 올렸다면 그 코드가 프로젝트의 가장 최신 버전일 것이다. 모든 코드를 자신이 가지고 있지 않아도 되고 깃허브에서 받아오기 때문에 메모리 측면에서도 유리하다. 그리고 branch 를 이용해서 첫 소스코드로부터 새로운 branch(비선형적 개발)들을 생성하고 여러 사람이 동시에 그 코드를 접근하고 변경할 수 있다.(자신들의 전문분야에만 더욱 집중적으로 개발할 수 있다.) 즉 분산적으로 개발이 가능하다. 이 후 merge과정을 통해 코드를 병합하여 다시 개발을 할 수 있다.(코드가 적절하다고 판단하면 다시 branch로 여러 명이 개발 할 것이다.) 그리고 개발 도중 에러가 발생하더라도 이전 branch로 쉽게 변경 가능하여 에러 처리에도 유용하고 이전 코드와 쉽게 비교가 가능하다.(commit 으로 수정 코드 확정) 즉 한사람으로 인해 프로젝트 코드가 망가질 걱정이 줄어들고 협업에 더욱 유용할 것이다.